priority encoder.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
    PORT ( w  : IN   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           y  : OUT  STD_LOGIC_VECTOR(1 DOWNTO 0) ;
           z  : OUT  STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
    WITH w SELECT
        y <=  "00" WHEN "0001",
              "01" WHEN "0010",
              "01" WHEN "0011",
              "10" WHEN "0100",
              "10" WHEN "0101",
              "10" WHEN "0110",
              "10" WHEN "0111",
              "11" WHEN OTHERS ;
    WITH w SELECT
        z <=  '0' WHEN "0000",
              '1' WHEN OTHERS ;
END Behavior ;
```

Selected Signal Assignment

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
    PORT ( w  : IN   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           y  : OUT  STD_LOGIC_VECTOR(1 DOWNTO 0) ;
           z  : OUT  STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
    y <= "11" WHEN w(3) = '1' ELSE
         "10" WHEN w(2) = '1' ELSE
         "01" WHEN w(1) = '1' ELSE
         "00" ;
    z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

Conditional Signal Assignment

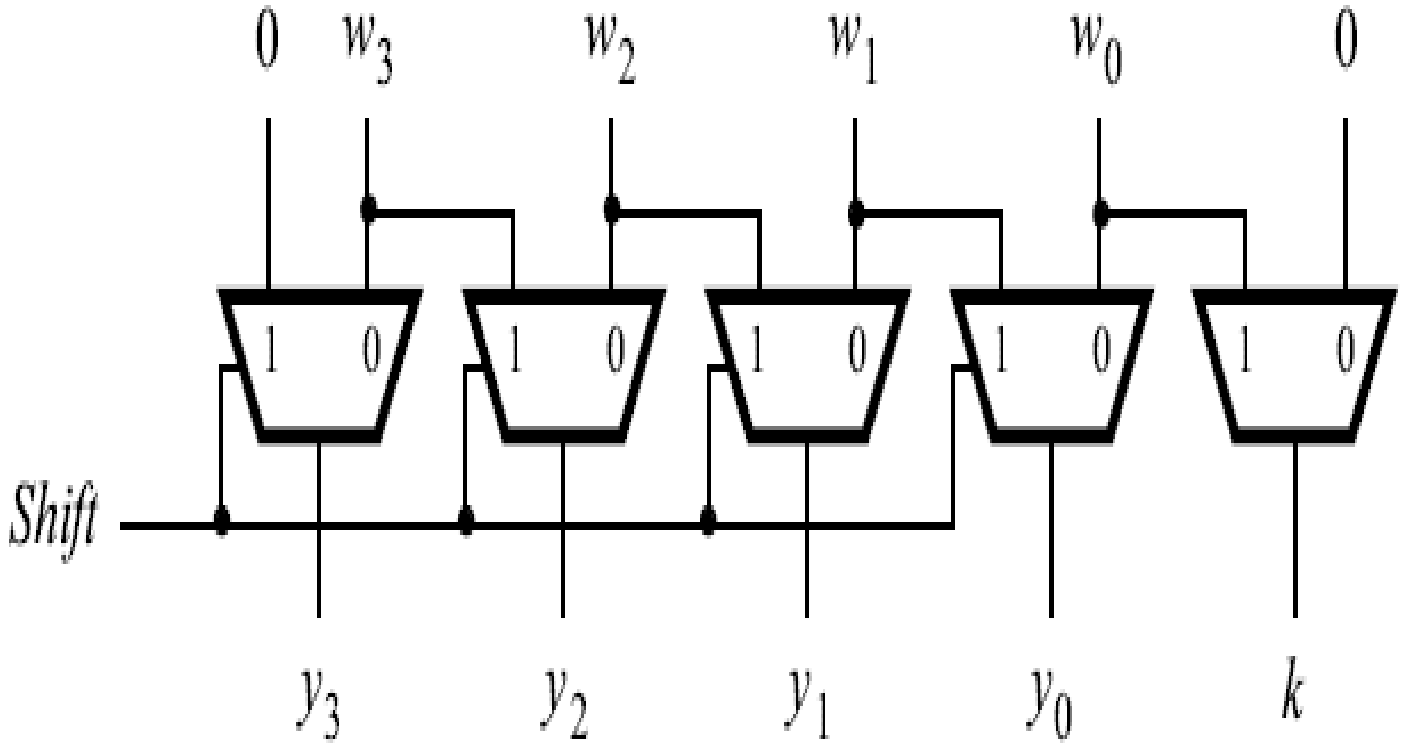
Hierarchical code for a 16-to-1 multiplexer using 4-to-1 mux

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
LIBRARY work ;
USE work.mux4to1_package.all ;

ENTITY mux16to1 IS
    PORT ( w  : IN   STD_LOGIC_VECTOR(0 TO 15) ;
           s  : IN   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           f  : OUT  STD_LOGIC ) ;
END mux16to1 ;

ARCHITECTURE Structure OF mux16to1 IS
    SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;
BEGIN
    Mux1: mux4to1 PORT MAP
        ( w(0), w(1), w(2), w(3), s(1 DOWNTO 0), m(0) ) ;
    Mux2: mux4to1 PORT MAP
        ( w(4), w(5), w(6), w(7), s(1 DOWNTO 0), m(1) ) ;
    Mux3: mux4to1 PORT MAP
        ( w(8), w(9), w(10), w(11), s(1 DOWNTO 0), m(2) ) ;
    Mux4: mux4to1 PORT MAP
        ( w(12), w(13), w(14), w(15), s(1 DOWNTO 0), m(3) ) ;
    Mux5: mux4to1 PORT MAP
        ( m(0), m(1), m(2), m(3), s(3 DOWNTO 2), f ) ;
END Structure ;
```

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.mux4to1_package.all ;

ENTITY mux16to1 IS
    PORT ( w  : IN   STD_LOGIC_VECTOR(0 TO 15) ;
           s  : IN   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           f  : OUT  STD_LOGIC ) ;
END mux16to1 ;

ARCHITECTURE Structure OF mux16to1 IS
    SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;
BEGIN
    G1: FOR i IN 0 TO 3 GENERATE
        Muxes: mux4to1 PORT MAP (
            w(4*i), w(4*i+1), w(4*i+2), w(4*i+3), s(1 DOWNTO 0), m(i) ) ;
    END GENERATE ;
    Mux5: mux4to1 PORT MAP ( m(0), m(1), m(2), m(3), s(3 DOWNTO 2), f ) ;
END Structure ;
```

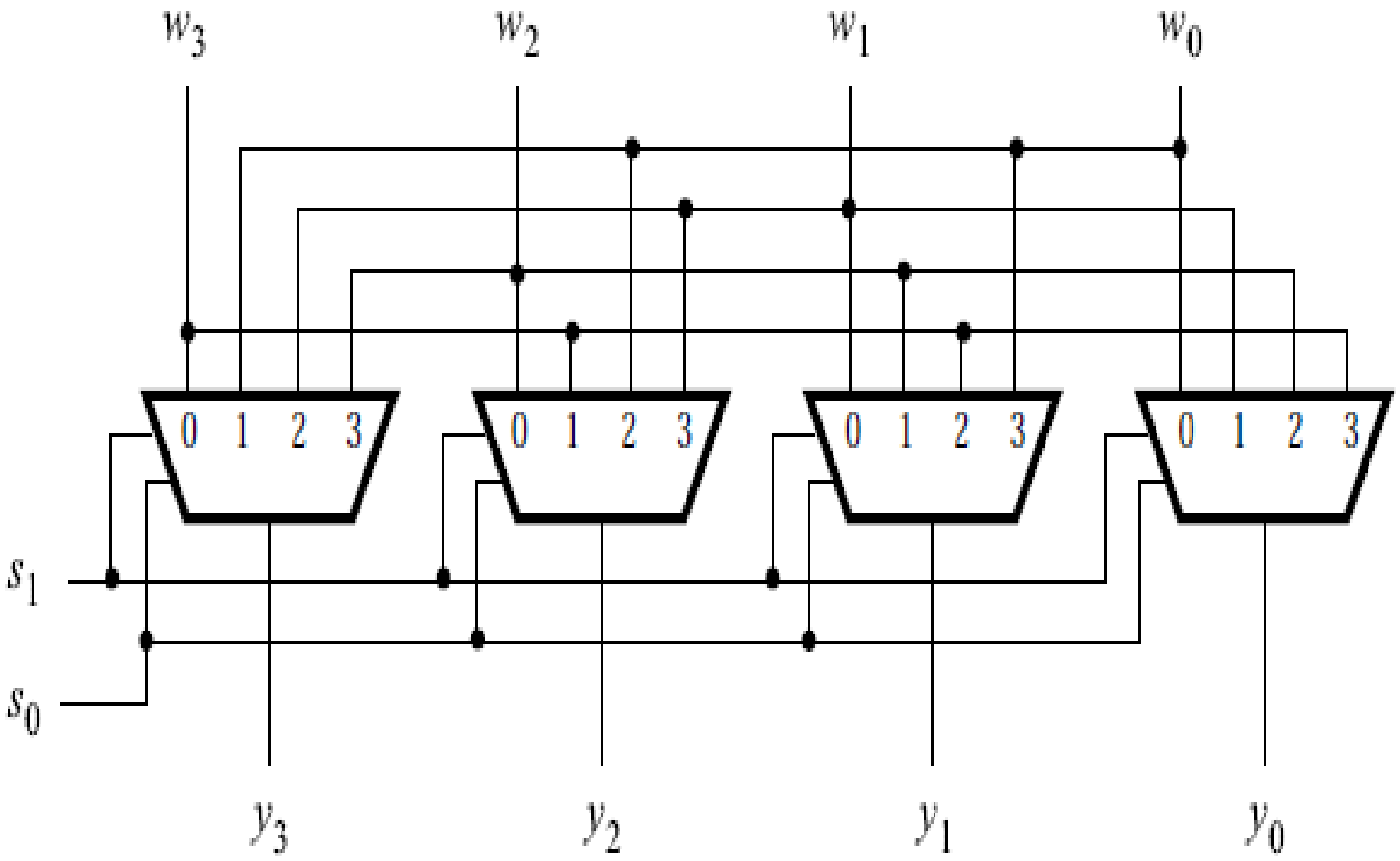
Hierarchical code for a 4-to-16 binary decoder.

```
Library ieee;
use     ieee.std_logic_1164.all;
Library work;
use     work.dec2to4_package.all;

entity dec4to16 is
      Port (
            w : in STD_LOGIC_vector(3 downto 0);
            En : in STD_LOGIC;
            y : out STD_LOGIC_vector(0 to 15)
                                    );
end dec4to16;

architecture structure of dec4to16 is
    signal m : STD_LOGIC_vector(0 upto 3);
    Begin
    Dec1 : dec2to4 Port map(w(3), w(2), En, m);

    G1 : For i IN 0 to 3 generate
      DEC : dec2to4 Port map

    ( w(1), w(0), m(i), y(4*i), y(4*i+1), y(4*i+2)
                        , y(4*i+3)
                                    );

    end generate;
end architecture;
```


0
w_3
w_2
w_1
w_0
0
1
0
1
0
1
0
1
0
1
0
Shift
y_3
y_2
y_1
y_0
k

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY shifter IS
    PORT ( w      : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           Shift  : IN  STD_LOGIC ;
           y      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           k      : OUT STD_LOGIC ) ;
END shifter ;

ARCHITECTURE Behavior OF shifter IS
BEGIN
    PROCESS (Shift, w)
    BEGIN
        IF Shift = '1' THEN
            y(3) <= '0' ;
            y(2 DOWNTO 0) <= w(3 DOWNTO 1) ;
            k <= w(0) ;
        ELSE
            y <= w ;
            k <= '0' ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY shifter IS
      PORT ( w      : IN     UNSIGNED(3 DOWNTO 0) ;
             Shift  : IN     STD_LOGIC ;
             y      : OUT    UNSIGNED(3 DOWNTO 0) ;
             k      : OUT    STD_LOGIC ) ;
END shifter ;

ARCHITECTURE Behavior OF shifter IS
BEGIN
      PROCESS (Shift, w)
      BEGIN
            IF Shift = "1" THEN
                  y <= w SRL 1 ;
                  k <= w(0) ;
            ELSE
                  y <= w ;
                  k <= "0" ;
            END IF ;
      END PROCESS ;
END Behavior ;
```


w_3
w_2
w_1
w_0
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3
s_1
s_0
y_3
y_2
y_1
y_0

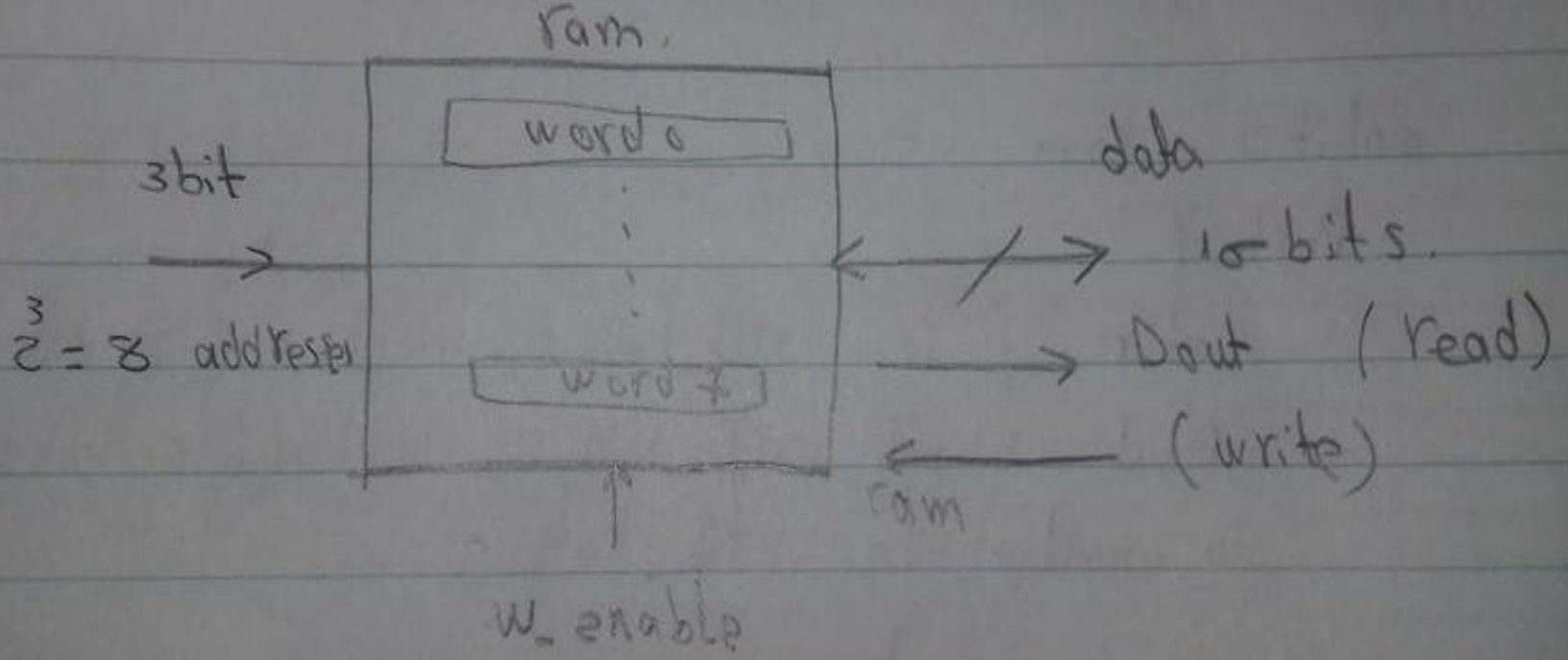
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY barrel IS
      PORT ( w   : IN     UNSIGNED(3 DOWNTO 0) ;
             s   : IN     UNSIGNED(1 DOWNTO 0) ) ;
             y   : OUT    UNSIGNED(3 DOWNTO 0) ) ;
END barrel ;

ARCHITECTURE Behavior OF barrel IS
BEGIN
      PROCESS (s, w)
      BEGIN
            CASE s IS
                  WHEN "00" =>
                        y <= w ;
                  WHEN "01" =>
                        y <= w ROR 1 ;
                  WHEN "10" =>
                        y <= w ROR 2 ;
                  WHEN OTHERS =>
                        y <= w ROR 3 ;
            END CASE ;
      END PROCESS ;
END Behavior ;
```

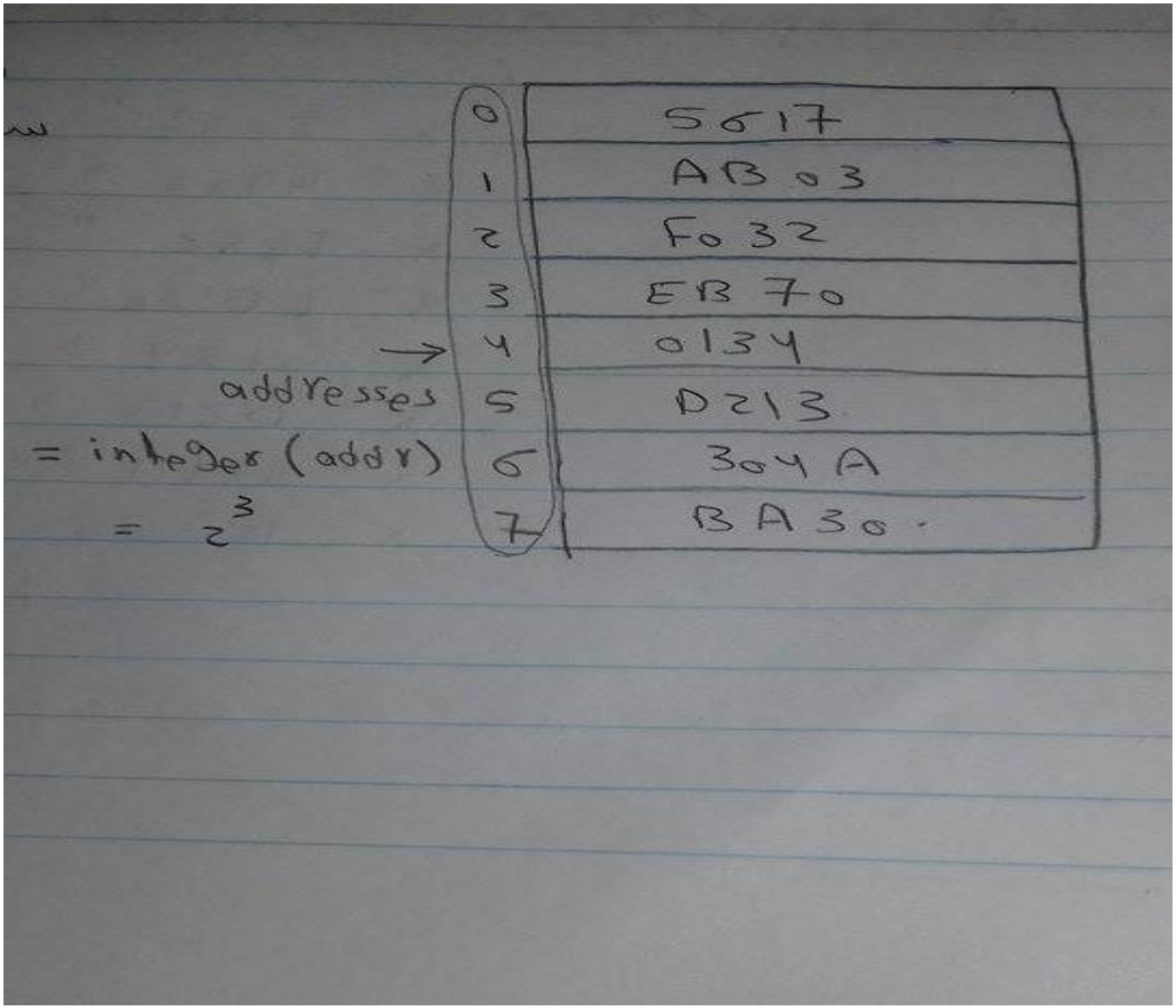
Rom

```
Library  ieee ;
use      ieee. STD_LOGIC_1164. all ;
use      ieee. STD_LOGIC_ unsigned ;

 entity  Rom  is
        Port (
           Addr :   in STD_LOGIC_Vector (2 downto 0);
           Dout :   out STD_LOGIC_Vector(15 downto 0)  ;
         );
 end  rom ;

architecture dataflow  of  rom  is
  signal  Temp : integer range 0 to 7 ;
 TYPE  rom_array is  array (0 to 7)  of  STD_LOGIC_Vector (15 downto 0);
   constant  memory  :  rom_array  : =
                        (
                          X" 2ceA"  ,
                          X " 0459"  ,
                          x " A412 "  ,
                          X " 0598 "  ,
                          X " CDAF "  ,
                          x " F1a3  "  ,
                          X " 5317 "  ,
                          x " 0a13   " ) ;

 BeGin
  Temp  <=   to_ integer (adder) ;
   Dout  <=  memory (temp) ;
end  data flow.
```

RAM

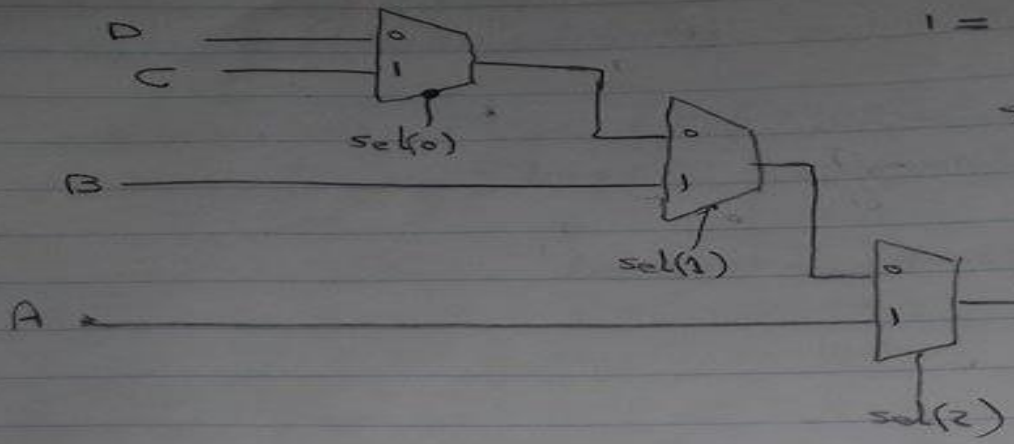
نفس طريقة RoM ولكن هاعل read & write.




addresses
= integer (addr)
= 2^3
0
1
2
3
4
5
6
7
5617
AB03
F032
EB70
0134
D213
304A
BA30.

```
library ieee;
use ieee.STD_LOGIC_1164.all;
use ieee.STD_LOGIC_unsigned;

entity ram is
  Port (
        addr : in STD_LOGIC_vector (2 downto 0);
        Dout : out STD_LOGIC_vector(15 downto 0);
        w_enable : in STD_LOGIC
        );
  end ram;
architecture dataflow of ram is
  signal temp : integer range 0 to 7;
  type ram_array is array (0 to 7) of STD_LOGIC_vector
                                      (15 downto 0);
  constant memory : ram_array := (
                                  x"5617",
                                  x"AB03",
                                  x"F032",
                                  x"EB70",
                                  x"0154",
                                  x"D213",
                                  x"304A",
                                  x"BA30");
Begin
```

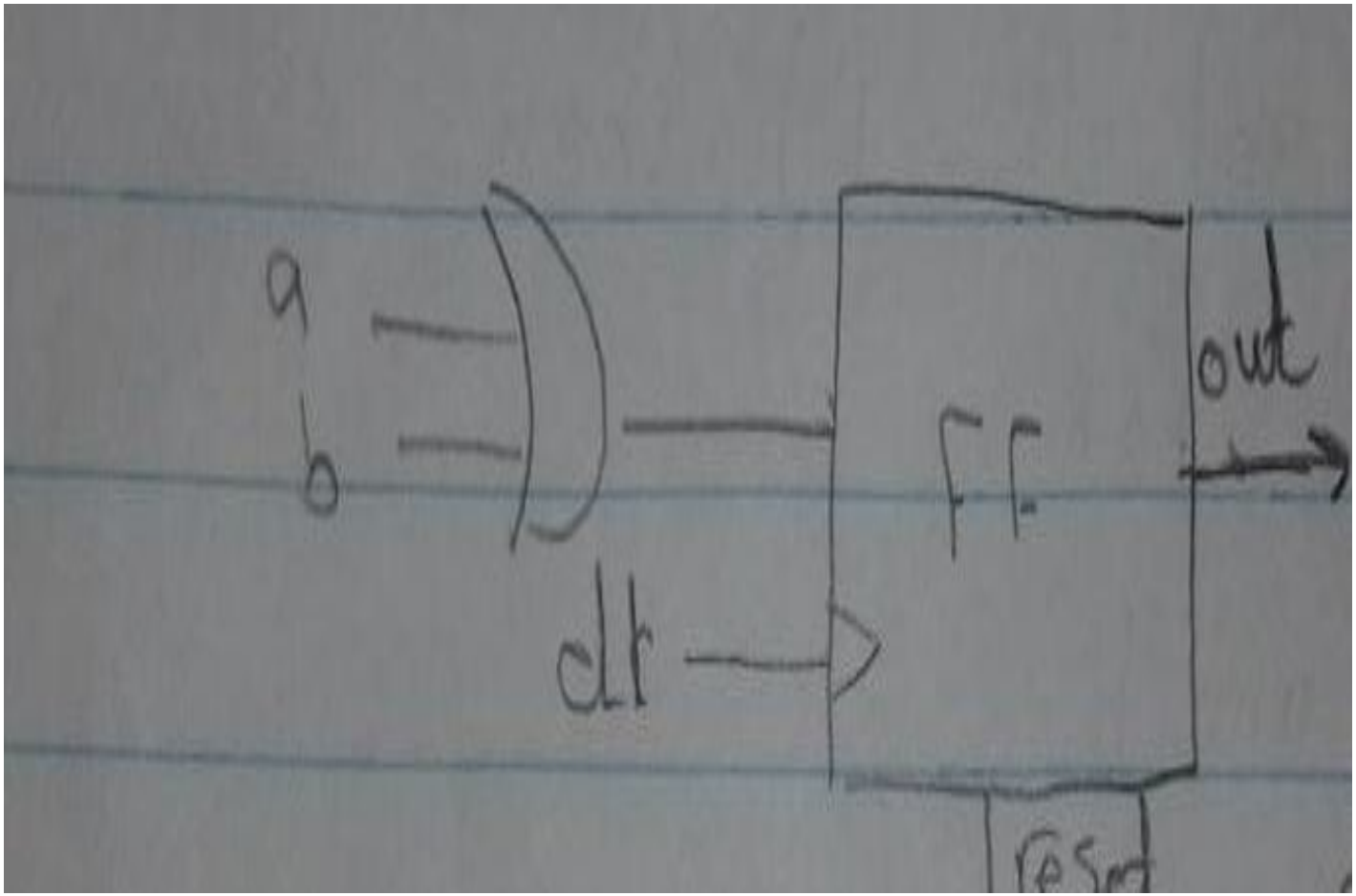
```
process ( addr , w_enable , Temp ).
    begin
        Temp <= to_integer ( addr ).

        if ( w_enable = '0' ) then.
            Dout <= memory ( Temp ) ;
        else
            memory (Temp) <= Dout ;
        endif.
    end process
end dataflow;
```

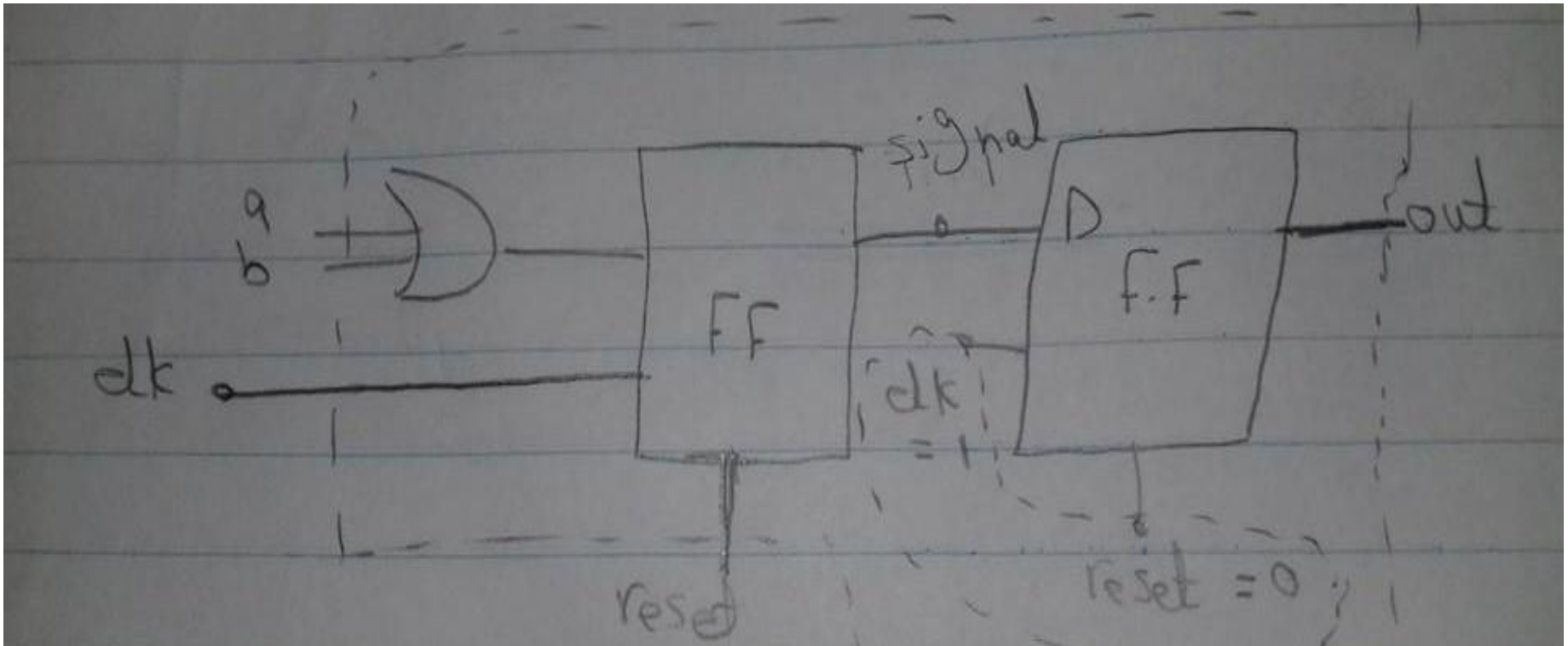



```
process ( sel , a , b , c , d )
  begin
if  ( sel(2) = '1' ) then
      y <= A ;
else if ( sel(1) = '1' ) then
      y <= B ;
else if ( sel(0) = '1' ) then
      y <= C ;
 else
      y <= D ;
end if ;
end process ;
```

ex

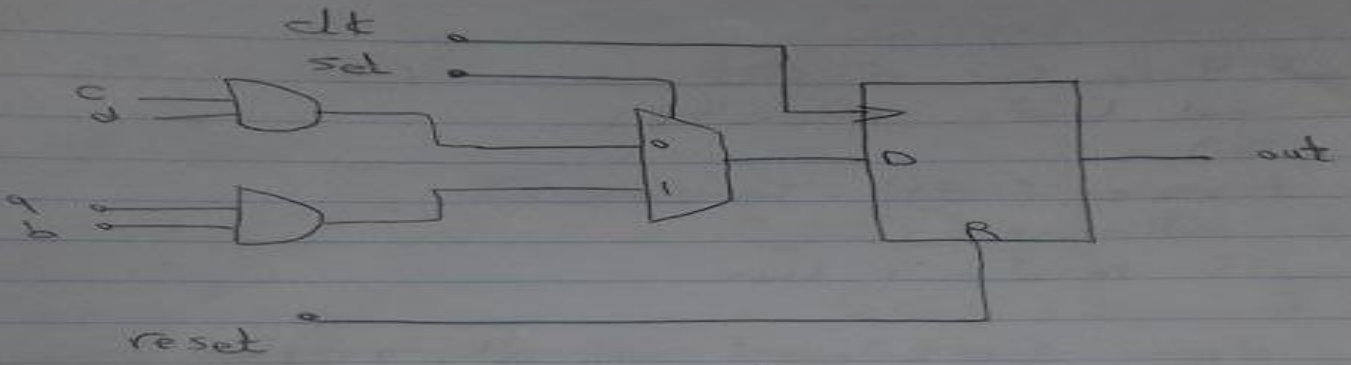
```
architecture var of FF is
  signal temf: std_logic;
  begin
  process (clk, Temf, a, b)
   begin
  if reset = '1' then
     out <= '0';
  elseif (clk'event and clk = '1') then
    temf <= a or b;
     out <= temf;
  end process;
end var;
```


a
b
clk
out



. Temp الى هساوى D داخلة out اذا

ex




```
process ( clk, reset, sel)
   begin
   if ( reset = '1') then
      out <= '0';
   else if ( clk'event and clk = '1') then
      if ( sel = '1') then
         out <= a and b;
      else
         out <= c and d;
      end if;
   end if;
   end process;
```